Computer Programming and Practice (Ⅰ)
計算機程式設計與實習(一)
- Ch06 -

110年上學期
國立臺南大學 電機工程系
梁家銘

6 陣　列 (Array)

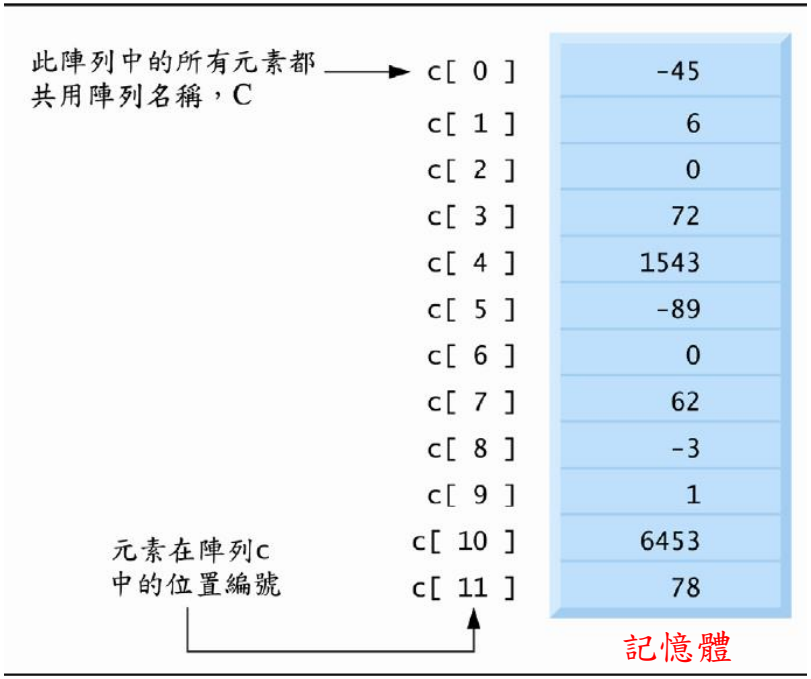
6.1 簡介

- **陣列 (Arrays)** 是由相同型別的資料組成的資料結構(data structure)。
  - 後面章節，將討論C語言的**struct**(結構)—可由不同型別的相關資料項所組成的資料結構。
  - 陣列和結構都屬於「靜態」的資料結構，在程式執行期間的大小不會改變。

6.2 陣列

- 陣列是一群具有相同型別的連續記憶體位置。
  - 若要存取陣列的某個元素，須指定陣列名稱、及此元素在陣列中的**位置編號 (position number)**。

- 宣告方式：
  陣列型態 陣列名稱[陣列長度];
  例如：int c[12]; //長度為12，整數型態



- 圖6.1表示一個名稱為c的整數陣列(含有12個元素)

- 陣列存取方式：
  - 陣列名稱後方，使用中括號並加入編號，如：c[5] ，則可存取指定元素，此編號亦稱為**下標 (subscript)**或**索引(index)**，須為整數或整數運算式 (如：c[x+y])。
  - **注意**：陣列元素編號，是由零開始編號，其稱為**零元素（zeroth element，** 如：c[0] ），陣列**c**的第**i**個元素則是**c[i-1]**。
  - 陣列名稱命名規則，如同一般變數名稱，僅能用字母、數字和底線，名稱不能以數字開頭。
  - 中括號亦是運算子，優先權較高，如右表所示

運算子							結合性	型別
[]	()	++ *(postfix)*	-- *(postfix)*				由左至右	最高
+	-	!	++ *(prefix)*	-- *(prefix)*	*(type)*		由右至左	一元
*		/	%				由左至右	乘法
+		-					由左至右	加法
<		<=	>	>=			由左至右	關係
==		!=					由左至右	相等
&&							由左至右	邏輯 AND
||							由左至右	邏輯 OR
?:							由右至左	條件
=		+=	-=	*=	/=	%=	由右至左	指定
,							由左至右	逗號

6.3 陣列定義(宣告)

- 陣列需佔用記憶體空間。
  - 陣列需指定元素型別和元素個數，電腦會依此來預留適當大小的記憶體空間。

1. 陣列宣告

```
int b[ 100 ], x[ 27 ];
```

- 上述宣告一個整數陣列**b**，並預留100個元素位置，及一個整數陣列**x**，並預留27個元素位置。
  - 此二矩陣分別有0~99與0~26的陣列編號(亦稱：索引/下標)。
  - 註：size_t 型別為無號整數型別，定義於<stdio.h>

6.4 陣列存取

2. 定義陣列並用迴圈初始化元素

- 右圖使用一個for迴圈，將10個元素的整數陣列**n**的初始為零，並印出。

(練習Trace code)

程式碼

fig06_03.c

```
1  // Fig. 6.3: fig06_03.c
2  // Initializing the elements of an array to zeros.
3  #include <stdio.h>
4
5  // function main begins program execution
6  int main(void)
7  {                          //長度為5的陣列 (含5個元素)
8     int n[5]; // n is an array of five integers
9
10    // set elements of array n to 0        //初始化為0
11    for (size_t i = 0; i < 5; ++i) {
                             //(i = 0, 1, 2, ..., 4)
12       n[i] = 0; // set element at location i to 0
13    }
14
15    printf("%s%13s\n", "Element", "Value");
16
17    // output contents of array n in tabular format
18    for (size_t i = 0; i < 5; ++i) {
19       printf("%7u%13d\n", i, n[i]);
20    }                                   //印出陣列
21 }
```

輸出

```
Element         Value
      0             0
      1             0
      2             0
      3             0
      4             0
```

圖6.3 將陣列所有元素的初始化為零

3. 以初始值串列來初始化陣列

- 陣列可在宣告時設定初始值，在宣告後加上等號和大括號(={ , , , })，並填上串陣列**初始值(array initializers)**
  - 例如：int n[5] = {32, 27, 64, 18, 95};
  - 下圖在宣告時為陣列設定5個初始值，並以表格方式列出陣列內容。

(逗號區隔)

fig06_04.c

```
1  // Fig. 6.4: fig06_04.c
2  // Initializing the elements of an array with an initializer list.
3  #include <stdio.h>
4
5  // function main begins program execution
6  int main(void)
7  {
8     // use initializer list to initialize array n
9     int n[5] = {32, 27, 64, 18, 95};
10
11    printf("%s%13s\n", "Element", "Value");
12
13    // output contents of array in tabular format
14    for (size_t i = 0; i < 5; ++i) {
15       printf("%7u%13d\n", i, n[i]);
16    }
17 }
```

//初始值

//初始化

//for迴圈印出: i, n[i]

輸出

```
Element        Value
      0           32
      1           27
      2           64
      3           18
      4           95
```

圖6.4　以初始值串列將陣列的元素初始化

4. 使用符號常數來指定陣列大小，並以計算來初始化陣列

- 下圖將5元素的陣列s之元素，設定初始值為2、4、6、8、10，並列表方式印出陣列。
  - 其中，陣列元素之內容值，是以迴圈計數器 i×2再+2取得。
  - 註：#define為前置處理器命令 (preprocessor directive)，用於定義符號常數 (symbolic constant)，並在程式編譯前，代換成對應內容，例如：
    #define SIZE 5　　//不用加分號

fig06_05.c

D:\D槽\Dropbox\109上 程式設計\教科書(光碟)_中文\教學投影片\中文版教學投影片\06146_ch6\06146_ch06程式碼\fig06_04.

```
1  //
2  // Initializing the elements of array s to the even integers from 2 to 20.
3  #include <stdio.h>
4  #define SIZE 5 // maximum size of array
5        //定義常數
6  // function main begins program execution
7  int main(void)
8  {
9     // symbolic constant SIZE can be used to specify array size
10    int s[SIZE]; // array s has SIZE elements
11          //陣列元素設值
12    for (size_t j = 0; j < SIZE; ++j) { // set the values
13       s[j] = 2 + 2 * j;          //s[0] = 2,
14    }                                 s[1] = 4, …, s[4] = 10.
15
16    printf("%s%13s\n", "Element", "Value");
17
18    // output contents of array s in tabular format
19    for (size_t j = 0; j < SIZE; ++j) {
20       printf("%7u%13d\n", j, s[j]); //印出s[0], s[1], …, s[4]
21    }
22 }
```

程式碼

輸出

Element	Value
0	2
1	4
2	6
3	8
4	10

圖6.5　將陣列元素初始化成2到20的偶數

5. 計算陣列元素值之總和

- 下圖將12元素的整數陣列a之所有內容值利用for迴圈進行加總。
  - 注意：若初始值個數少於陣列元素個數，則剩餘元素自動設為0。

fig06_06.c

程式碼

```
1  // Fig. 6.6: fig06_06.c
2  // Computing the sum of the elements of an array.
3  #include <stdio.h>
4  #define SIZE 12
5          //定義常數
6  // function main begins program execution
7  int main(void)
8  {
9     // use an initializer list to initialize the array   //陣列宣告，同時初始化
10    int a[SIZE] = { 1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45 };
11    int total = 0; // sum of array
12
13    // sum contents of array a                  //陣列元素加總
14    for (size_t i = 0; i < SIZE; ++i) {
15       total += a[i];    //意同：total = total + a[i];
16    }
17
18    printf("Total of array element values is %d\n", total);
19 }
```

輸出

```
Total of array element values is 383
```

圖6.6　計算陣列所有元素值之總和

6. 應用：使用陣列來加總調查結果

- 以下利用陣列整理問卷資料，問題描述如下：
  - 請40位學生對自助餐廳評分(分數為1分~10分: 1分代表差，10分代表非常好)。
  - 將此40份回應資料存放至整數陣列中，並統計每種評分(1分至10分) 之數量。

fig06_07.c

```
1  // Fig. 6.7: fig06_07.c
2  // Analyzing a student poll.
3  #include <stdio.h>
4  #define RESPONSES_SIZE 40 // define array sizes //回應筆數
5  #define FREQUENCY_SIZE 11 //出現次數 (0分跳過)
6
7  // function main begins program execution
8  int main(void)
9  {
10    // initialize frequency counters to 0
11    int frequency[FREQUENCY_SIZE] = {0}; //初始化陣列 (剩餘值全補為0)
12
13    // place the survey responses in the responses array
14    int responses[RESPONSES_SIZE] = {1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
15       1, 6, 3, 8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6,
16       5, 6, 7, 5, 6, 4, 8, 6, 8, 10};      //40次評分之內容
17
18    // for each answer, select value of an element of array responses
19    // and use that value as an index in array frequency to
20    // determine element to increment          //統計每種評分之次數
21    for (size_t answer = 0; answer < RESPONSES_SIZE; ++answer) {
22       ++frequency[responses[answer]];
23    } //意即：frequency[responses[answer]] = frequency[responses[answer]] +1 ;
24       // Ex: frequency[5] = frequency[5] +1; //5分的次數加1
25    // display results
26    printf("%s%17s\n", "Rating", "Frequency");
27
28    // output the frequencies in a tabular format      //印出結果
29    for (size_t rating = 1; rating < FREQUENCY_SIZE; ++rating) {
30       printf("%6d%17d\n", rating, frequency[rating]);
31    }
32 }
```

輸出

Rating	Frequency
1	2
2	2
3	2
4	2
5	5
6	11
7	5
8	7
9	1
10	3

圖6.7　學生意見調查分析程式

7. 以長條圖表示陣列元素值

- 下例從陣列讀取數值，並將數值以長條圖 (histogram) 印出
  - 先印出每個數字，再根據其數值大小印出對應星號數*。

fig06_08.c

```
1  // Fig. 6.8: fig06_08.c
2  // Displaying a histogram.
3  #include <stdio.h>
4  #define SIZE 5
5        //定義常數
6  // function main begins program execution
7  int main(void)
8  {
9     // use initializer list to initialize array n
10    int n[SIZE] = {19, 3, 15, 7, 11}; //初始化內容
11
12    printf("%s%13s%17s\n", "Element", "Value", "Histogram");
13
14    // for each element of array n, output a bar of the histogram
15    for (size_t i = 0; i < SIZE; ++i) {
16       printf("%7u%13d          ", i, n[i]) ;
17
18       for (int j = 1; j <= n[i]; ++j) { // print one bar
19          printf("%c", '*');
20       }                          //針對每個數，透過for迴圈產生星號
21
22       puts(""); // end a histogram bar with a newline
23    }
24 }
```

輸出

```
Element        Value          Histogram
      0           19          *******************
      1            3          ***
      2           15          ***************
      3            7          *******
      4           11          ***********
```

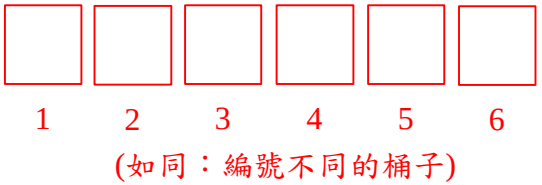
圖6.8　長條圖列印

8. 投擲骰子6,000,000次並將結果寫入陣列中

- 問題描述：
  - 投擲一個6面骰子6,000,000次，來驗證亂數產生器是否均勻。

fig06_09.c

```
1  // Fig. 6.9: fig06_09.c
2  // Roll a six-sided die 60,000,000 times
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <time.h>
6  #define SIZE 7
7        //定義常數
8  // function main begins program execution
9  int main(void)
10 {                                   //初始化陣列 (內容值全設為0)
11    unsigned int frequency[SIZE] = {0}; // clear counts
12
13    srand(time(NULL)); // seed random number generator
14                       //從1900年1月1日到現在的時間秒數
15    // roll die 60,000,000 times
16    for (unsigned int roll = 1; roll <= 60000000; ++roll) {
17       size_t face = 1 + rand() % 6;
18       ++frequency[face]; // replaces entire switch of Fig. 5.12
19    }
                  //統計各種點數出現次數，如：++frequency[3]; //點數3，增加一次 (帶值進去跑一次，有助於了解)
20
21    printf("%s%17s\n", "Face", "Frequency");
22
23    // output frequency elements 1-6 in tabular format
24    for (size_t face = 1; face < SIZE; ++face) {
25       printf("%4d%17d\n", face, frequency[face]);
26    }
27 }
```



(如同：編號不同的桶子)

輸出

```
Face        Frequency
   1           999753
   2          1000773
   3           999600
   4           999786
   5          1000552
   6           999536
```

//接近1百萬次

圖6.9　使用陣列 (取代**switch**的擲骰子)

6.5 使用字元陣列來儲存及操作字串

程式碼

- 下例以字串常數作為字元陣列初始值，讀入一個字串到字元陣列中，以字串方式印出字元陣列，並存取字串中的個別字元等。
  - 註：每個字串都包含終止字元 (string-termination character)，或稱為空字元(null character)，字元以 '\0' 作為表示。

fig06_10.c

```
1  // Fig. 6.10: fig06_10.c
2  // Treating character arrays as strings.
3  #include <stdio.h>
4  #define SIZE 20
5        //定義常數
6  // function main begins program execution
7  int main(void)
8  {                    //宣告字元型態陣列(即字串)
9     char string1[SIZE]; // reserves 20 characters
10    char string2[] = "string literal"; // reserves 15 characters
11                   //以字串("string literal")作為初始值，並決定陣列長度(14+1)
12    // read string from user into array string1
13    printf("%s", "Enter a string (no longer than 19 characters): ");
14    scanf("%19s", string1); // input no more than 19 characters
15       //以%s 讀入字串，變數前不必加 '&' (限制最多寫入19個字元，若讀到空格，則會停止)
16    // output strings
17    printf("string1 is: %s\nstring2 is: %s\n"
18           "string1 with spaces between characters is:\n",
19           string1, string2);
20
21    // output characters until null character is reached
22    for (size_t i = 0; i < SIZE && string1[i] != '\0'; ++i) {
23       printf("%c ", string1[i]);
24    }                          //for迴圈，逐一印出字元or空格，直到'\0'
25
26    puts("");          //插入空格
27 }
```

//註：string1 沒寫括弧，亦表示 &string1[0]的意思

輸出

```
Enter a string (no longer than 19 characters): Hello there
string1 is: Hello
string2 is: string literal
string1 with spaces between characters is:
H e l l o
```

圖6.10 將字元陣列視為字串

6.6 靜態區域陣列以及自動區域陣列

- 以下例子，比較static陣列及一般陣列：

1. 在**staticArrayInit**函式中宣告**static**陣列
   - **static**陣列會在程式執行期間一直保存
   - **static**陣列初始值自動設為零
   - 函式二次呼叫時，static陣列保有上次數值。

2. 在**automaticArrayInit**函式，宣告一般陣列。
   - 函式二次呼叫時，一般陣列不會保留上次數值。

fig06_11.c

```
1  // Fig. 6.11: fig06_11.c
2  // Static arrays are initialized to zero if not
   explicitly initialized.
3  #include <stdio.h>
4
5  void staticArrayInit(void); // function prototype
6  void automaticArrayInit(void); // function prototype
7
8  // function main begins program execution
9  int main(void)
10 {
11    puts("First call to each function:");
12    staticArrayInit();
13    automaticArrayInit();              //第一回合
14
15    puts("\n\nSecond call to each function:");
16    staticArrayInit();
17    automaticArrayInit();              //第二回合
18 }
19
20 // function to demonstrate a static local array
21 void staticArrayInit(void)
22 {
23    // initializes elements to 0 before the function
      is called
24    static int array1[3];              //第1個陣列：宣告為static陣列(初始值自動設為0)
25
26    puts("\nValues on entering staticArrayInit:");
27
28    // output contents of array1       //for迴圈將陣列設值
29    for (size_t i = 0; i <= 2; ++i) {
30       printf("array1[%u] = %d  ", i, array1[i]);
31    }
32
33    puts("\nValues on exiting staticArrayInit:");
```

```
34
35      // modify and output contents of array1
36      for (size_t i = 0; i <= 2; ++i) {
37         printf("array1[%u] = %d  ", i, array1[i] += 5);
38      }
39   }
40
41   // function to demonstrate an automatic local array
42   void automaticArrayInit(void)
43   {
44      // initializes elements each time function is
         called
45      int array2[3] = { 1, 2, 3 };
46
```

//第2個陣列 (長度為3，區域變數)

```
47      puts("\n\nValues on entering automaticArrayInit:");
48
49      // output contents of array2
50      for (size_t i = 0; i <= 2; ++i) {
51         printf("array2[%u] = %d  ", i, array2[i]);
52      }
53
54      puts("\nValues on exiting automaticArrayInit:");
55
56      // modify and output contents of array2
57      for (size_t i = 0; i <= 2; ++i) {
58         printf("array2[%u] = %d  ", i, array2[i] += 5);
59      }
60   }
```

//印出陣列內容

//修改陣列內容

輸出

```
First call to each function:

Values on entering staticArrayInit:
array1[0] = 0  array1[1] = 0  array1[2] = 0
Values on exiting staticArrayInit:
array1[0] = 5  array1[1] = 5  array1[2] = 5

Values on entering automaticArrayInit:
array2[0] = 1  array2[1] = 2  array2[2] = 3
Values on exiting automaticArrayInit:
array2[0] = 6  array2[1] = 7  array2[2] = 8

Second call to each function:
Values on entering staticArrayInit:
array1[0] = 5  array1[1] = 5  array1[2] = 5
Values on exiting staticArrayInit:
array1[0] = 10  array1[1] = 10  array1[2] = 10

Values on entering automaticArrayInit:
array2[0] = 1  array2[1] = 2  array2[2] = 3
Values on exiting automaticArrayInit:
array2[0] = 6  array2[1] = 7  array2[2] = 8
```

//第一回合

//第二回合

//static陣列(保留上次值)

values preserved from last call

//一般陣列

values reinitialized after last call

圖6.11　若未明確指定初始值，static陣列的初始值將自動設為零

6.7 傳遞陣列給函式

- 若想傳遞陣列至函式，僅需指定陣列名稱，不必加中括號，如：function(ArrayName);
  - 記住：陣列名稱也代表陣列第一個元素之位址。
  - C語言傳遞陣列給函式會以傳參考(call-by-reference, 亦即傳位址)，因此函式可修改陣列內容。
  - 下例以%p轉換指定詞 (用以印出位址)，分別印出**array**、**&array[0]**和**&array**之位址，以說明陣列名稱即代表陣列第一個元素之位址。

fig06_12.c

```
1  // Fig. 6.12: fig06_12.c
2  // Array name is the same as the address of the array   first
   element.
3  #include <stdio.h>
4
5  // function main begins program execution
6  int main(void)
7  {
8     char array[5]; // define an array of size 5
9
10                                  //%p: 印出位址
   printf("    array = %p\n&array[0] = %p\n   &array = %p\n",
11       array, &array[0], &array);
12 }
```

```
    array = 0012FF78
&array[0] = 0012FF78
   &array = 0012FF78
```

圖6.12 陣列名稱和陣列第一個元素之位址為相同

1. 傳遞整個陣列與傳遞陣列元素之差異

- 下例示範傳遞整個陣列和傳遞陣列單一元素之差異。

程式碼

fig06_13.c

```c
// Fig. 6.13: fig06_13.c
// Passing arrays and individual array elements to functions.
#include <stdio.h>
#define SIZE 5 //定義常數

// function prototypes
void modifyArray(int b[], size_t size); //修改整個陣列
void modifyElement(int e);    //修改陣列單一元素

// function main begins program execution
int main(void)
{
   int a[SIZE] = { 0, 1, 2, 3, 4 }; // initialize array a

   puts("Effects of passing entire array by reference:\n\nThe "
      "values of the original array are:");

   // output original array
   for (size_t i = 0; i < SIZE; ++i) {
      printf("%3d", a[i]);
   }

   puts(""); // outputs a newline
                  //傳遞整個陣列，僅寫陣列名稱(不必加括號)
   modifyArray(a, SIZE); // pass array a to modifyArray by
   reference              //記得告知陣列長度(否則函式不知道多長)
   puts("The values of the modified array are:");

   // output modified array
   for (size_t i = 0; i < SIZE; ++i) {
      printf("%3d", a[i]);//印出陣列內容
   }

   // output value of a[3]
   printf("\n\n\nEffects of passing array element "
      "by value:\n\nThe value of a[3] is %d\n", a[3]);

   modifyElement(a[3]); // pass array element a[3] by value
                  //傳遞陣列單一元素，如同傳遞變數(函式
   // output value of a[3]          只複製變數內容，無法修改)
   printf("The value of a[3] is %d\n", a[3]);
}
```

```c
// in function modifyArray, "b" points to the original array
"a"
// in
memory                    //陣列引數，中括號為空，不指定大小
void modifyArray(int b[], size_t size)
{
   // multiply each array element by 2   //for迴圈改值(乘2)
   for (size_t j = 0; j < size; ++j) {
      b[j] *= 2; // actually modifies original
      array
   }
}

// in function modifyElement, "e" is a local copy of array
element
// a[3] passed from main
void modifyElement(int e)
{
   // multiply parameter by 2
   printf("Value in modifyElement is %d\n", e *= 2);
}
```

輸出

```
Effects of passing entire array by reference:

The values of the original array are:
  0  1  2  3  4
The values of the modified array are:
  0  2  4  6  8

Effects of passing array element by value:

The value of a[3] is 6
Value in modifyElement is 12
The value of a[3] is 6
```

//傳遞整個陣列，內容值可被修改

//傳遞陣列單一元素至函式，內容值無法修改

圖6.13　傳遞陣列和各別的陣列元素給函式

2. 陣列參數使用const修飾詞

- 下列說明**const**修飾詞用法
  - 函式**tryToModifyArray**夾帶參數**const int b[];**定義，表示陣列**b**為常數型態(禁止修改)。
  - 若不當修改，編譯器會產生錯誤訊息。

fig06_14.c

```
1  // Fig. 6.14: fig06_14.c
2  // Using the const type qualifier with arrays.
3  #include <stdio.h>
4
5  void tryToModifyArray(const int b[]); // function prototype
6
7  // function main begins program execution
8  int main(void)
9  {          //不指定長度，由設值個數決定
10    int a[] = { 10, 20, 30 }; // initialize array a
11
12    tryToModifyArray(a);
13
14    printf("%d %d %d\n", a[0], a[1], a[2]);
15 }
16
17 // in function tryToModifyArray, array b is const, so it cannot be
18 // used to modify its argument array in the caller.
19 void tryToModifyArray(const int b[]) //const常數宣告(禁止修改內容)
20 {
21    b[0] /= 2; // error
22    b[1] /= 2; // error   //一旦修改，編譯器會跳出error!(禁止修改)
23    b[2] /= 2; // error
24 }
```

圖6.14　陣列使用const限制修改

6.8 陣列的排序

- 排序 (Sorting)資料
  - 按照遞增或遞減順序是程式最普遍的應用之一。
- 右例示範**氣泡排序(bubble sort或sinking sort)**方法，將含有10個元素的陣列a 進行遞增排序。
  - 氣泡排序－較小數值會如「氣泡」浮上水面(移至陣列前端)，而較大數值則會沉到陣列尾端。
  - 容易撰寫，但效率較差。
  - *舉例：9, 5, 2, 7, 4

畫黑板

```
Data items in original order
   2   6   4   8  10  12  89  68  45  37
Data items in ascending order
   2   4   6   8  10  12  37  45  68  89
```
//由小到大排序

輸出

圖6.15　由大到小來排序一個陣列

fig06_15.c

```
1  // Fig. 6.15: fig06_15.c
2  // Sorting an array's values into ascending order.
3  #include <stdio.h>
4  #define SIZE 10 //定義常數(數值為10)
5
6  // function main begins program execution
7  int main(void)
8  {
9     // initialize a                          //未經排序之數列
10    int a[SIZE] = {2, 6, 4, 8, 10, 12, 89, 68, 45, 37};
11
12    puts("Data items in original order");
13
14    // output original array
15    for (size_t i = 0; i < SIZE; ++i) {
16       printf("%4d", a[i]);
17    }
18
19    // bubble sort
20    // loop to control number of passes          //氣泡排序
21    for (unsigned int pass = 1; pass < SIZE; ++pass) {
22
23       // loop to control number of comparisons per pass
24       for (size_t i = 0; i < SIZE - 1; ++i) {
25
26          // compare adjacent elements and swap them if first
27          // element is greater than second element
28          if (a[i] > a[i + 1]) {
29             int hold = a[i];
30             a[i] = a[i + 1];
31             a[i + 1] = hold;
32          }
33       }
34    }
35
36    puts("\nData items in ascending order");
37
38    // output sorted array
39    for (size_t i = 0; i < SIZE; ++i) {
40       printf("%4d", a[i]);
41    }
42
43    puts("");
44 }
```
//注意：Swap (兩數交換)：
a[i] ⇆ a[i+1] 需暫借一變數借放

//找工作的面試常考 (排序)

6.9 範例：用陣列計算平均數、中位數及眾數

//中間編號的數字

//出現最多次的數字

- **調查資料分析 (survey data analysis)**，統計問卷調查結果。
  - 下圖使用陣列**response**來初始化99筆問卷回應，每筆數值範圍都介於1~9之間。
  - 此程式將計算99筆數值之平均數、中位數和眾數。

fig06_16.c

```c
// Fig. 6.16: fig06_16.c
// Survey data analysis with arrays;
// computing the mean, median and mode of the data.
#include <stdio.h>
#define SIZE 99 //定義一個常數

// function prototypes //常數引數(不可修改)
void mean(const unsigned int answer[]);
void median(unsigned int answer[]); //取中位數
void mode(unsigned int freq[], const unsigned int answer[]) ; //眾數
void bubbleSort(unsigned int a[]);
void printArray(const unsigned int a[]);

// function main begins program execution
int main(void)
{
   unsigned int frequency[10] = {0}; // initialize array frequency

   // initialize array response
   unsigned int response[SIZE] = //問卷結果(1~9分)
      {6, 7, 8, 9, 8, 7, 8, 9, 8, 9,
       7, 8, 9, 5, 9, 8, 7, 8, 7, 8,
       6, 7, 8, 9, 3, 9, 8, 7, 8, 7,
       7, 8, 9, 8, 9, 8, 9, 7, 8, 9,
       6, 7, 8, 7, 8, 7, 9, 8, 9, 2,
       7, 8, 9, 8, 9, 8, 9, 7, 5, 3,
       5, 6, 7, 2, 5, 3, 9, 4, 6, 4,
       7, 8, 9, 6, 8, 7, 8, 9, 7, 8,
       7, 4, 4, 2, 5, 3, 8, 7, 5, 6,
       4, 5, 6, 1, 6, 5, 7, 8, 7};

   // process responses
   mean(response);
   median(response);
   mode(frequency, response);  //main函式很乾淨 (老闆呼叫員工)
}
```

```c
// calculate average of all response values
void mean(const unsigned int answer[])
{
   printf("%s\n%s\n%s\n", "********", "  Mean", "********");

   unsigned int total = 0; // variable to hold sum of array elements

   // total response values //統計總分
   for (size_t j = 0; j < SIZE; ++j) {
      total += answer[j];  //意即：total = total + answer[j];
   }

   printf("The mean is the average value of the data\n"
          "items. The mean is equal to the total of\n"
          "all the data items divided by the number\n"
          "of data items (%u). The mean value for\n"
          "this run is: %u / %u = %.4f\n\n",
          SIZE, total, SIZE, (double) total / SIZE); //計算平均
}

// sort array and determine median element's value
void median(unsigned int answer[])
{
   printf("\n%s\n%s\n%s\n%s",
          "********", " Median", "********",
          "The unsorted array of responses is");

   printArray(answer); // output unsorted array
   //排序(氣泡排序)，數字由小到大，找中間數
   bubbleSort(answer); // sort array

   printf("%s", "\n\nThe sorted array is");
   printArray(answer); // output sorted array
```

```
73      printf("\n\nThe median is element %u of\n"
74             "the sorted %u element array.\n"
75             "For this run the median is %u\n\n",
76             SIZE / 2, SIZE, answer[SIZE / 2]);
77   }
78
79   // determine most frequent response
80   void mode(unsigned int freq[], const unsigned int answer[])
81   {
82      printf("\n%s\n%s\n%s\n", "********", "  Mode", "********");
83
84      // initialize frequencies to 0
85      for (size_t rating = 1; rating <= 9; ++rating) {
86         freq[rating] = 0;                    //如同：9種不同編號的箱子
87      }
88
89      // summarize frequencies                //統計評分出現次數
90      for (size_t j = 0; j < SIZE; ++j) {
91         ++freq[answer[j]]; //意即：freq[answer[j]] = freq[answer[j]] + 1;
92      }
93
94      // output headers for result columns
95      printf("%s%11s%19s\n\n%54s\n%54s\n\n",
96             "Response", "Frequency", "Histogram",
97             "1    1    2    2", "5    0    5    0    5");
98                                              //長條核度
99      // output results
100     unsigned int largest = 0; // represents largest frequency
101     unsigned int modeValue = 0; // represents most frequent response
102
103     for (size_t rating = 1; rating <= 9; ++rating) {
104        printf("%8u%11u          ", rating, freq[rating]);
105
106        // keep track of mode value and largest frequency value
107        if (freq[rating] > largest) {
108           largest = freq[rating]; //最大值
109           modeValue = rating; //最高次數
110        }
111
112        // output histogram bar representing frequency value
113        for (unsigned int h = 1; h <= freq[rating]; ++h) {
114           printf("%s", "*");
115        }                                    //畫出*狀長條圖
116
117        puts(""); // being new line of output
118     }
119
120     // display the mode value
121     printf("\nThe mode is the most frequent value.\n"
122            "For this run the mode is %u which occurred"
123            " %u times.\n", modeValue, largest);
124  }                          //最高次數   //最大值
```

```
125
126  // function that sorts an array with bubble sort algorithm
127  void bubbleSort(unsigned int a[])
128  {
129     // loop to control number of passes
130     for (unsigned int pass = 1; pass < SIZE; ++pass) {
131
132        // loop to control number of comparisons per pass
133        for (size_t j = 0; j < SIZE - 1; ++j) {
134
135           // swap elements if out of order
136           if (a[j] > a[j + 1]) {
137              unsigned int hold = a[j];
138              a[j] = a[j + 1];
139              a[j + 1] = hold;
140           }
141        }
142     }
143  }
144
145  // output array contents (20 values per row)
146  void printArray(const unsigned int a[])
147  {
148     // output array contents
149     for (size_t j = 0; j < SIZE; ++j) {
150
151        if (j % 20 == 0) { // begin new line every 20 values
152           puts("");
153        }                                    //每20筆，換行
154
155        printf("%2u", a[j]);
156     }
157  }
```

輸出

```
********
  Mean
********
The mean is the average value of the data
items. The mean is equal to the total of
all the data items divided by the number
of data items ( 99 ). The mean value for
this run is: 681 / 99 = 6.8788 //平均結果


********
 Median
********
The unsorted array of responses is
 6 7 8 9 8 7 8 9 8 9 7 8 9 5 9 8 7 8 7 8
 6 7 8 9 3 9 8 7 8 7 7 8 9 8 9 8 9 7 8 9
 6 7 8 7 8 7 9 8 9 2 7 8 9 8 9 8 9 7 5 3
 5 6 7 2 5 3 9 4 6 4 7 8 9 6 8 7 8 9 7 8
 7 4 4 2 5 3 8 7 5 6 4 5 6 1 6 5 7 8 7
```

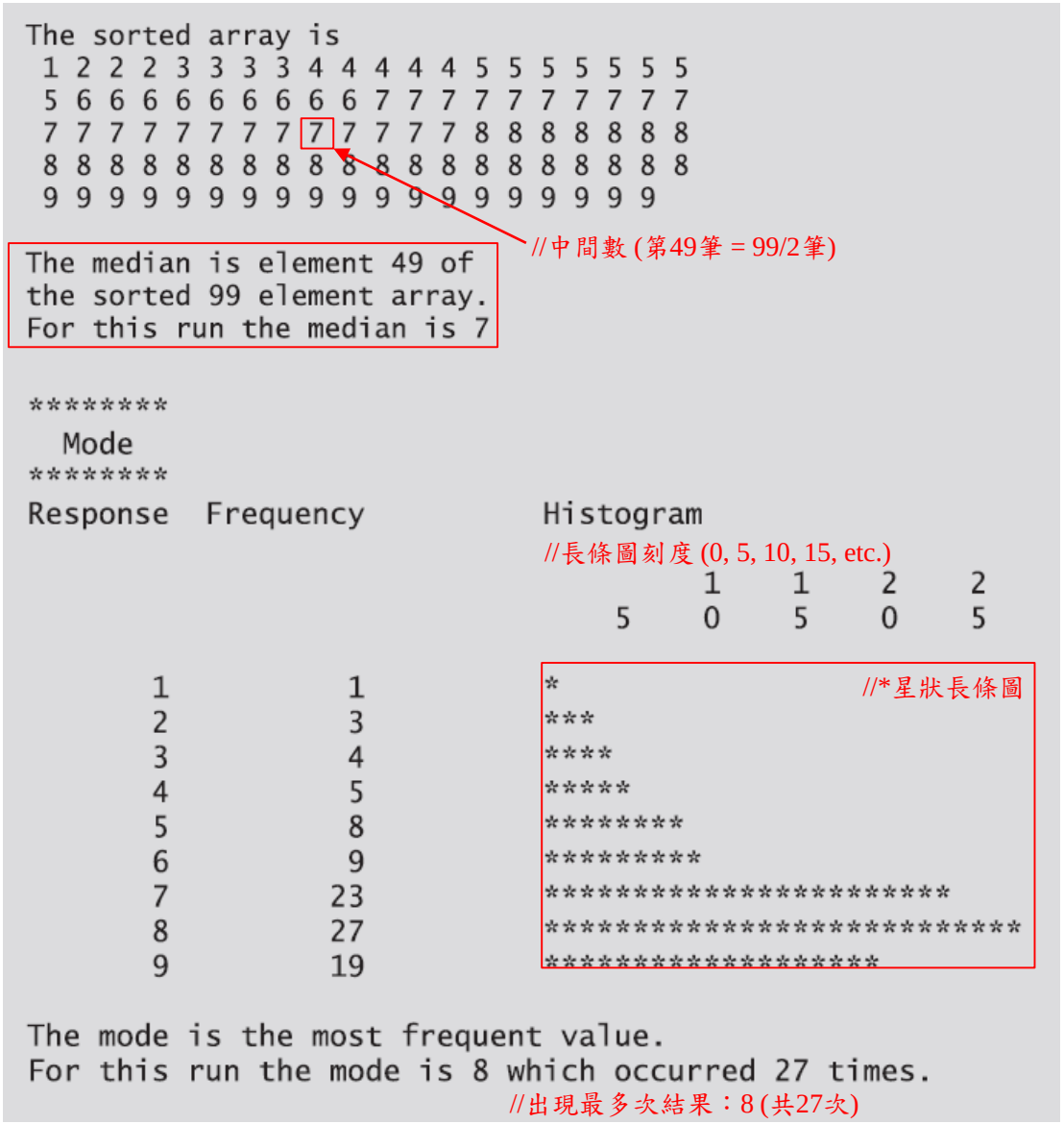


圖6.17　問卷資料之統計結果

1. 平均數(Mean)

- 平均數：算術平均數。
  - 函式**mean** 將99筆數值相加，再除99計算。

2. 中位數(Median)

- 中位數是「位於中間順序之數值」。
  - 函式 **median** 呼叫 **bubbleSort** (氣泡排序)進行遞增排序，然後選出位於中間之數值 **answer[SIZE/2]**。

3. 眾數(Mode)

- 眾數：出現頻率最高之數值。
  - 函式**mode** 對每種評分統計次數，並找到出現次數最多之結果。

6.10 搜尋陣列 (Searching Arrays)

- **搜尋(searching)**：當在陣列中存放許多資料，想從陣列找尋符合特定的**關鍵值 (key value)** 數值。

1. **使用線性搜尋(Linear Search)來搜尋陣列**

- 線性搜尋：逐筆比較陣列中之每筆元素是否符合。
  - 註：由於陣列資料雜亂，因此可能很快找到，或可能最後一筆元素才找到。
  - 平均而言，程式需要搜尋一半陣列元素。

```
Enter integer search key:
36
Found value in element 18
```

輸出

```
Enter integer search key:
37
Value not found
```

圖6.18 陣列的線性搜尋

fig06_18.c

```c
// Fig. 6.18: fig06_18.c
// Linear search of an array.
#include <stdio.h>
#define SIZE 100

// function prototype                                  //傳遞陣列至函式
size_t linearSearch(const int array[], int key, size_t size);

// function main begins program execution
int main(void)
{
   int a[SIZE]; // create array a

   // create some data                                 //陣列設值：0, 2, 4, 6, .. 198.
   for (size_t x = 0; x < SIZE; ++x) {
      a[x] = 2 * x;
   }

   printf("Enter integer search key: ");
   int searchKey; // value to locate in array a
   scanf("%d", &searchKey); //讀取指定值(搜尋用)

   // attempt to locate searchKey in array a
   size_t index = linearSearch(a, searchKey, SIZE);

   // display results
   if (index != -1) {
      printf("Found value at index %d\n", index);
   }
   else {
      puts("Value not found");
   }
}

// compare key to every element of array until the location is found
// or until the end of array is reached; return index of element
// if key is found or -1 if key is not
found
size_t linearSearch(const int array[], int key, size_t size)
{
   // loop through array                               //for迴圈，逐一比較
   for (size_t n = 0; n < size; ++n) {


      if (array[n] == key) {
         return n; // return location of key
      }
   }

   return -1; // key not found
}
```

2. 使用二元搜尋 (Binary Search) 來搜尋陣列

- 線性搜尋對於小型陣列或未排序的陣列表現較好。
  - 但不適合大型陣列

- 若陣列數值已排序，則可採用更快的二元搜尋法(binary search)。
  - 舉例：1, 2, 3, 4, 5, 6, 7, 8, 9
  - 搜尋"6" (5→6 就找到!)
  - 思考：序列長n，共要比較幾次? 答: 至少x次 ($2^x \geq n$)

- 右圖為**binarySearch**函式之迭代/迴圈版本(習題有遞迴版本)，此函式接收四個引數：
  1. 整數陣列**b**
  2. **searchKey** (搜尋目標)
  3. 陣列的**low** (起點編號)
  4. 陣列的**high** (終點編號)

fig06_19.c

```
1  // Fig. 6.19: fig06_19.c
2  // Binary search of a sorted array.
3  #include <stdio.h>
4  #define SIZE 15
5
6  // function prototypes                                          //搜尋函式
7  size_t binarySearch(const int b[], int searchKey, size_t low, size_t
    high);                          //陣列   //搜尋值(常數) //起始編號  //終止編號
8  void printHeader(void);
9  void printRow(const int b[], size_t low, size_t mid, size_t high);
10
11 // function main begins program execution
12 int main(void)
13 {
14    int a[SIZE]; // create array a
15
16    // create data          //陣列設定數值(0, 2, 4, 6, … 28, 由小至大)
17    for (size_t i = 0; i < SIZE; ++i) {
18       a[i] = 2 * i;
19    }
20
21    printf("%s", "Enter a number between 0 and 28: ");
22    int key; // value to locate in array a
23    scanf("%d", &key); //取得：欲搜尋之數值
24
25    printHeader();
                  //印出標號及-----
26
27    // search for key in array a                  //呼叫二元搜尋函式
28    size_t result = binarySearch(a, key, 0, SIZE - 1);
29
30    // display results
31    if (result != -1) {
32       printf("\n%d found at index %d\n", key, result);
33    }
34    else {
35       printf("\n%d not found\n", key);
36    }
37 }
38
39 // function to perform binary search of an array
40 size_t binarySearch(const int b[], int searchKey, size_t low, size_t
    high)
41 {
42    // loop until low index is greater than high index
43    while (low <= high) {
                     //確保起始編號，小於終止編號
44       // determine middle element of subarray being searched
45       size_t middle = (low + high) / 2;
46
47       // display subarray used in this loop iteration
48       printRow(b, low, middle, high); //印出此段陣列內容
49
```

```
50
51          // if searchKey matched middle element, return middle
52          if (searchKey == b[middle]) {
53             return middle;
54          }
55
56          // if searchKey is less than middle element, set new high
57          else if (searchKey < b[middle]) {
58             high = middle - 1; // search low end of array
59          }
60
61          // if searchKey is greater than middle element, set new low
62          else {
63             low = middle + 1; // search high end of array
64          }
65       } // end while
66
67       return -1; // searchKey not found
68    }
69
70    // Print a header for the output
71    void printHeader(void)
72    {
73       puts("\nSubscripts:");
74
75       // output column head
76       for (unsigned int i = 0; i < SIZE; ++i) {
77          printf("%3u ", i);
78       }
79
80       puts(""); // start new line of output
81
82       // output line of - characters
83       for (unsigned int i = 1; i <= 4 * SIZE; ++i) {
84          printf("%s", "-");
85       }
86
87       puts(""); // start new line of output
88    }
89
90    // Print one row of output showing the current
91    // part of the array being processed.
92    void printRow(const int b[], size_t low, size_t mid, size_t high)
93    {
94       // loop through entire array
95       for (size_t i = 0; i < SIZE; ++i) {
96
```

//比較小：位於前段，更新終止編號

//比較大：位於後段，更新起始編號

//印出該段陣列

```
97          // display spaces if outside current subarray range
98          if (i < low || i > high) { //若超出範圍：印空白
99             printf("%s", "    ");
100         }
101         else if (i == mid) { // display middle element
102            printf("%3d*", b[i]); // mark middle value
103         }          //中間值，加顆星
104         else { // display other elements in subarray
105            printf("%3d ", b[i]);
106         }
107      }
108
109      puts(""); // start new line of output
110   }
```

輸出

```
Enter a number between 0 and 28: 25

Subscripts:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
---------------------------------------------------------------
  0   2   4   6   8  10  12  14* 16  18  20  22  24  26  28
                                 16  18  20  22* 24  26  28
                                                 24  26* 28
                                                 24*

25 not found
```

//中間值 //不斷縮小範圍(每次減半)

```
Enter a number between 0 and 28: 8

Subscripts:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
---------------------------------------------------------------
  0   2   4   6   8  10  12  14* 16  18  20  22  24  26  28
  0   2   4   6*  8  10  12
                  8  10* 12
                  8*

8 found in array element 4
```

```
Enter a number between 0 and 28: 6

Subscripts:
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
---------------------------------------------------------------
  0   2   4   6   8  10  12  14* 16  18  20  22  24  26  28
  0   2   4   6*  8  10  12

6 found in array element 3
```

圖6.19　在已排序的陣列中進行二元搜尋

6.11 多維陣列 (類似多維矩陣)

- C語言的陣列可以有多重下標/索引。
  - **多重下標/索引陣列 (multiple-subscripted arrays)**，亦稱為**多維陣列 (multidimensional arrays)** 常用於**表格(tables)**呈現，其數值是依列 (rows) 和行 (columns) 排列組成
  - 當下標為兩項，則為**二維陣列 (double-subscripted arrays)**，如下圖。

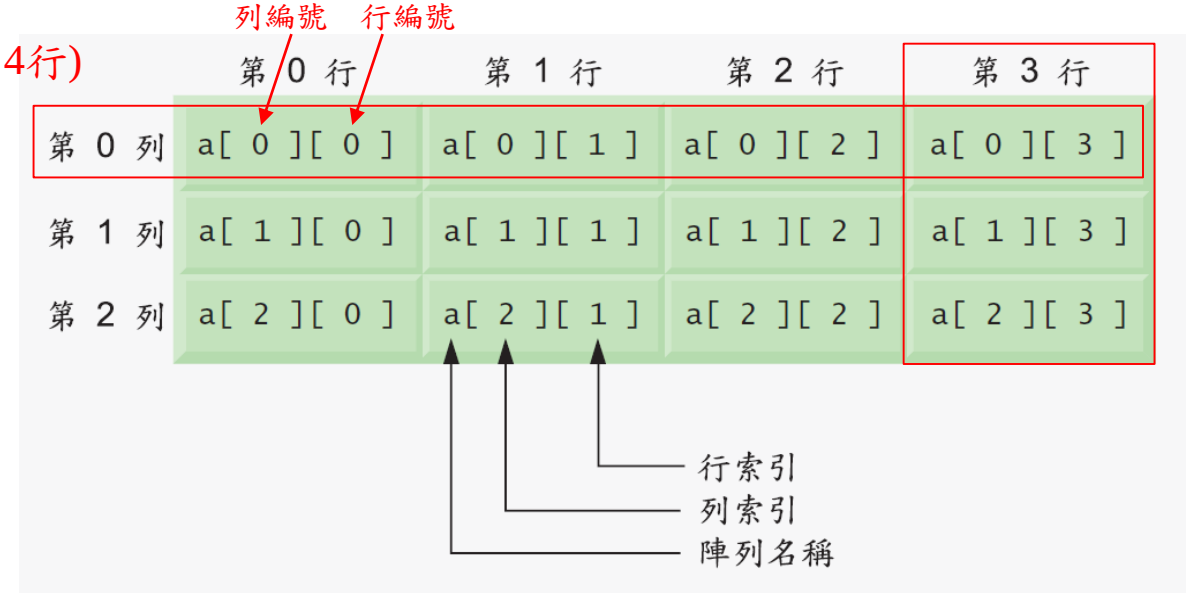


圖6.20　有三列以及四行的多重下標陣列

- 註：若下標/索引超過兩項，則形成多維陣列。

- 多維陣列，可在宣告時設定初始值：

fig06_21.c

```c
// Fig. 6.21: fig06_21.c
// Initializing multidimensional arrays.
#include <stdio.h>

                                //傳入陣列，第一個下標不需指定(其他需要)
void printArray(int a[][3]); // function prototype

// function main begins program execution
int main(void)
{                   //2列3行          //第一列          //第二列
   int array1[2][3] = { { 1, 2, 3 }, { 4, 5, 6 } };
   puts("Values in array1 by row are:");
   printArray(array1);

                                //可以用一列來代表，缺值自動補0
   int array2[2][3] = { 1, 2, 3, 4, 5 };
   puts("Values in array2 by row are:");
   printArray(array2);

                                //缺值自動補0
   int array3[2][3] = { { 1, 2 }, { 4 } };
   puts("Values in array3 by row are:");
   printArray(array3);
}

// function to output array with two rows and three columns
void printArray(int a[][3])
{
   // loop through rows //兩層for迴圈，印出2維陣列
   for (size_t i = 0; i <= 1; ++i) {

      // output column values
      for (size_t j = 0; j <= 2; ++j) {
         printf("%d ", a[i][j]);
      }

      printf("\n"); // start new line of output
   }
}
```

輸出

```
Values in array1 by row are:
1 2 3
4 5 6
Values in array2 by row are:
1 2 3
4 5 0
Values in array3 by row are:
1 2 0
4 0 0
```

- 傳遞多維陣列至函式時，參數陣列的第一個索引不需寫，但其他索引必須寫。
  - 無論是幾維陣列，所有的陣列元素都是連續存放在記憶體中。

1. 二維陣列的處理

- 下例為3位同學及4科成績進行統計，3×4陣列：每列代表一位學生，每行代表四次考試成績。
  - 函式minimum：找出最低成績。
  - 函式maximum：找出最高成績。
  - 函式average：求出平均成績。
  - 函式printArray：以列表方式印出陣列。

(練習自行trace code看看)

fig06_22.c

```c
// Fig. 6.22: fig06_22.c
// Two-dimensional array manipulations.
#include <stdio.h>
#define STUDENTS 3
#define EXAMS 4

// function prototypes
int minimum(const int grades[][EXAMS], size_t pupils, size_t tests
); //找最低成績
int maximum(const int grades[][EXAMS], size_t pupils, size_t tests
); //找最高成績
double average(const int setOfGrades[], size_t tests);//找平均成績
void printArray(const int grades[][EXAMS], size_t pupils, size_t
tests);//印出陣列內容

// function main begins program execution
int main(void)
{
   // initialize student grades for three students (rows)
   int studentGrades[STUDENTS][EXAMS] =
      { { 77, 68, 86, 73 },//第一位同學
        { 96, 87, 89, 78 },//第二位同學
        { 70, 90, 86, 81 } };//第三位同學

   // output array studentGrades
   puts("The array is:");
   printArray(studentGrades, STUDENTS, EXAMS);//印出陣列內容

   // determine smallest and largest grade values
   printf("\n\nLowest grade: %d\nHighest grade: %d\n",
      minimum(studentGrades, STUDENTS, EXAMS),
      maximum(studentGrades, STUDENTS, EXAMS));
```

// 3 x 4 陣列 (3位學生 x 4科成績)

```c

   // calculate average grade for each student
   for (size_t student = 0; student < STUDENTS; ++student) {
      printf("The average grade for student %u is %.2f\n",
         student, average(studentGrades[student], EXAMS));
   }
}

// Find the minimum grade
int minimum(const int grades[][EXAMS], size_t pupils, size_t tests)
{
   int lowGrade = 100; // initialize to highest possible grade
                       // 訂一個分數，若比它小，則更新
   // loop through rows of grades
   for (size_t i = 0; i < pupils; ++i) {

      // loop through columns of grades
      for (size_t j = 0; j < tests; ++j) {

         if (grades[i][j] < lowGrade) {
            lowGrade = grades[i][j];
         }
      }
   }

   return lowGrade; // return minimum grade
}

// Find the maximum grade// 找最大值
int maximum(const int grades[][EXAMS], size_t pupils, size_t tests)
{
   int highGrade = 0; // initialize to lowest possible grade
                      // 訂一個分數，若比它大，則更新
   // loop through rows of grades
   for (size_t i = 0; i < pupils; ++i) {

      // loop through columns of grades
      for (size_t j = 0; j < tests; ++j) {

         if (grades[i][j] > highGrade) {
            highGrade = grades[i][j];
         }
      }
   }

   return highGrade; // return maximum grade
}

// Determine the average grade for a particular student
double average(const int setOfGrades[], size_t tests) // 取平均
{
   int total = 0; // sum of test grades // 總分
```

```
81      int total = 0; // sum of test grades
82
83      // total all grades for one student
84      for (size_t i = 0; i < tests; ++i) {
85         total += setOfGrades[i]; //意即：total = total + setOfGrades[i];
86      }
87
88      return (double) total / tests; // average
89   }                                 //計算平均
90
91   // Print the array
92   void printArray(const int grades[][EXAMS], size_t pupils, size_t
     tests)
93   {
94      // output column heads
95      printf("%s", "                  [0]  [1]  [2]  [3]");
96
97      // output grades in tabular format     //印出陣列
98      for (size_t i = 0; i < pupils; ++i) {
99
100        // output label for row
101        printf("\nstudentGrades[%u] ", i);
102
103        // output grades for one student
104        for (size_t j = 0; j < tests; ++j) {
105           printf("%-5d", grades[i][j]);
106        }
107     }
108  }
```

輸出

```
The array is:
                 [0]  [1]  [2]  [3]
studentGrades[0] 77   68   86   73
studentGrades[1] 96   87   89   78
studentGrades[2] 70   90   86   81

Lowest grade: 68
Highest grade: 96
The average grade for student 0 is 76.00 //此同學的平均分
The average grade for student 1 is 87.50
The average grade for student 2 is 81.75
```

6.12 可變長度陣列 (Variable-Length Arrays)

- 即時宣告陣列長度 (VALs)：來處理未知需求大小陣列。
  - 非大小可變，是指長度(或大小)在程式執行階段才定義。
  - 右例：宣告並印出多筆可變長度陣列。
  - 註1：sizeof在執行時期運作。
  - 註2：可變長度陣列傳遞至函數和一般陣列相同。

fig06_23.c

```
1  // Fig. 6.23: fig06_23.c
2  // Using variable-length arrays in C99
3  #include <stdio.h>
4
5  // function prototypes
6  void print1DArray(size_t size, int array[size]);
7  void print2DArray(size_t row, size_t col, int array[row][col]);
8
9  int main(void)
10 {
11    printf("%s", "Enter size of a one-dimensional array: ");
12    int arraySize; // size of 1-D array
13    scanf("%d", &arraySize);
14                                   //一維陣列的長度，待 scanf 後得知
15    int array[arraySize]; // declare 1-D variable-length array
16
17    printf("%s", "Enter number of rows and columns in a 2-D array:
      ");
18    int row1, col1; // number of rows and columns in a 2-D array
19    scanf("%d %d", &row1, &col1); //第一組陣列：行、列大小
20
21    int array2D1[row1][col1]; // declare 2-D variable-length array
22
23    printf("%s",
24       "Enter number of rows and columns in another 2-D array: ");
25    int row2, col2; // number of rows and columns in another 2-D
      array
26    scanf("%d %d", &row2, &col2); //第二組陣列：行、列大小
27
28    int array2D2[row2][col2]; // declare 2-D variable-length array
29                                   //二維陣列的行、列，待執行才得知
30    // test sizeof operator on VLA
31    printf("\nsizeof(array) yields array size of %d bytes\n",
32       sizeof(array));//印出陣列大小
33
34    // assign elements of 1-D VLA     //寫入值: i²
35    for (size_t i = 0; i < arraySize; ++i) {
36       array[i] = i * i;
37    }
38
39    // assign elements of first 2-D VLA  //寫入陣列1之數值: i + j
40    for (size_t i = 0; i < row1; ++i) {
41       for (size_t j = 0; j < col1; ++j) {
42          array2D1[i][j] = i + j;
43       }
44    }
45
46    // assign elements of second 2-D VLA  //寫入陣列2之數值: i + j
47    for (size_t i = 0; i < row2; ++i) {
48       for (size_t j = 0; j < col2; ++j) {
49          array2D2[i][j] = i + j;
50       }
```

```
50          }
51      }
52
53      puts("\nOne-dimensional array:");
54      print1DArray(arraySize, array); // pass 1-D VLA to function
55
56      puts("\nFirst two-dimensional array:");
57      print2DArray(row1, col1, array2D1); // pass 2-D VLA to function
58
59      puts("\nSecond two-dimensional array:");
60      print2DArray(row2, col2, array2D2); // pass other 2-D VLA to
        function
61  }
62
63  void print1DArray(size_t size, int array[size])
64  {
65      // output contents of array          //印出一維陣列內容
66      for (size_t i = 0; i < size; i++) {
67          printf("array[%d] = %d\n", i, array[i]);
68      }
69  }
70
71  void print2DArray(size_t row, size_t col, int array[row][col])
72  {
73      // output contents of array          //印出二維陣列內容
74      for (size_t i = 0; i < row; ++i) {
75          for (size_t j = 0; j < col; ++j) {
76              printf("%5d", array[i][j]);
77          }
78
79          puts("");
80      }
81  }
```

輸出

```
Enter size of a one-dimensional array: 6
Enter number of rows and columns in a 2-D array: 2 5
Enter number of rows and columns in another 2-D array: 4 3

sizeof(array) yields array size of 24 bytes

One-dimensional array:
array[0] = 0
array[1] = 1
array[2] = 4
array[3] = 9
array[4] = 16
array[5] = 25

First two-dimensional array:
    0    1    2    3    4
    1    2    3    4    5

Second two-dimensional array:
    0    1    2
    1    2    3
    2    3    4
    3    4    5
```

//寫入值: i^2

//寫入 2 × 5 陣列: i + j

//寫入 4 × 3 陣列: i + j

圖6.23　執行時才決定陣度陣列

[工商服務]

分享：老師指導專題生-近期成果

- 【2021/11/25】
- 2021全國大專產學實作競賽
- 人工智慧及其應用組「全國第二名」
- 獎金：2萬元

- 【2021/12/08】
- 2021 創新創意創業競賽 (三創競賽)
- 創新創意組「全校第一名」
- 獎金：1萬元
  - 電機粉絲頁：https://reurl.cc/95jdRj